IMPLEMENTASI ALGORITMA NEGASCOUT PADA PERMAINAN ANIMAL CHESS

Sebastian Vincent S.[1] sbastianvincent7@gmail.com
Joko Purwadi, M.Kom.[2] jokop@ukdw.ac.id
Nugroho Agus H, M.Si.[3] cnuq@ukdw.ac.id

Abstract

Aplikasi permainan berbasis ponsel semakin beragam. Catur hewan merupakan permainan yang memanfaatkan logika kecerdasan buatan. Permainan ini mengandalkan kemampuan berpikir secara logis sehingga diperlukan algoritma yang memanfaatkan teknik pencarian untuk memainkan permainan ini.

Dalam penelitian ini, peneliti akan mengimplementasikan algoritma NegaScout. Algoritma NegaScout dapat memotong node–node yang tidak perlu, sehingga proses perhitungan dan pencarian node terbaik bisa dilakukan dengan cepat. Peneliti juga membandingkan dengan algoritma Minimax dan algoritma Alpha-Beta dalam hal jumlah node yang dihasilkan dan lamanya waktu yang dibutuhkan.

Melalui penelitian ini, disimpulkan bahwa agen komputer cerdas dengan menggunakan algoritma NegaScout membutuhkan waktu yang lebih cepat dan memotong node lebih banyak daripada algoritma Minimax dan Alpha-beta dalam mencari dan menemukan langkah terbaik dalam permainan Animal Chess.

Keywords: *komputer cerdas, animal chess, negascout*

1. Pendahuluan

Salah satu permainan yang memanfaatkan komputer cerdas adalah Animal Chess. Salah satu metode untuk pembuatan komputer cerdas pada Animal Chess adalah dengan menggunakan metode NegaScout. Metode ini sangat cocok digunakan untuk pembuatan computer cerdas dalam permainan yang memerlukan dua pemain. Dengan metode tersebut, kemungkinan yang ada adalah menang, seri atau kalah. Melalui penelitian tugas akhir ini, penulis diharapkan dapat membuat implementasi dari metode NegaScout dan metode evaluasi dalam permainan Animal Chess sehingga dapat dibuat sebuah aplikasi ponsel permainan Animal Chess dengan komputer cerdas yang memiliki kemampuan berpikir seperti manusia sebagai lawannya.

2. Dasar Teori

2.1. Komputer Cerdas

Kecerdasan buatan merupakan salah satu bagian ilmu komputer yang membuat komputer melakukan pekerjaan seperti dan sebaik yang dilakukan manusia. Kecerdasan buatan dikenalkan pertama kali oleh Warren McCulloch dan Walter Pitts pada tahun 1943.

[1]*Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*
[2]*Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*
[3]*Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*

Definisi dari kecerdasan buatan adalah cabang dari ilmu komputer terkait dengan belajar dan menciptakan sistem komputer yang mana memperlihatkan beberapa format dari kecerdasan. Sistem mempelajari konsep baru dan tugas, sistem dapat memberi reaksi dan solusi terbaik tentang dunia sekitar kita, sistem dapat mengerti bahasa alami atau merasa dan mengerti keadaan sekitar dengan *visual scene,* dan sistem melaksanakan pembuatan yang memerlukan kecerdasan manusia. Manusia sering menemui suatu masalah yang tidak dapat diselesaikan. Kecerdasan buatan membantu manusia untuk memecahkan permasalahan tersebut. Para peneliti melakukan berbagai penelitian untuk memecahkan permasalahan manusia dengan cara mensimulasikan cara kerja manusia ke dalam sistem. (Russell & Norvig, 1995)

2.2. Algoritma NegaScout

Negascout memiliki kesamaan dengan algoritma *Alpha-Beta*. Algoritma ini didasarkan pada ide pencarian di mana satu cabang pohon pertama dicari dan dievaluasi nilainya. Node ini kemudian akan diasumsikan memiliki nilai paling baik dari *tree* tersebut. Pencarian node yang lain bisa lebih cepat karena dengan asumsi nilai terbaik yang telah didapat maka dapat dilakukan pemotongan langsung terhadap cabang *tree* jika node selanjutnya memiliki nilai lebih rendah dari asumsi nilai terbaik. Jika ternyata ditemukan node dengan nilai yang lebih baik dari asumsi nilai terbaik, maka akan dilakukan pencarian kembali terhadap asumsi nilai terbaik. *Element* dasar yang digunakan dalam *NegaScout* mirip dengan *Alpha-Beta*. Jika posisi p adalah *leaf, NegaScout* mengembalikan nilai statisnya. Selain itu, variabel m dan n diinisialisasi dengan $-\infty$ dan beta. Kemudian *Negascout* mencari *successor* dari p dari kiri ke kanan. *Successor* paling kiri p.1 dicari dengan interval (-beta, -alpha) kemudian sisanya p.2, ... , p.b dicari dengan *zero-width window* (-m-1, -m) yang nilainya diberi sesaat setelah pencarian *successor* paling kiri.

```
1 FUNCTION negascout (p: POSITION; alpha, beta, depth: INTEGER) : INTEGER;
2 VAR i,t,m,n: INTEGER;
3 BEGIN
4     IF depth = d THEN RETURN (evaluate(p))
5     ELSE
6     BEGIN m := -∞;
7           n := beta;
8           FOR i := 1 TO b DO
9           BEGIN t := -negascout (p.i, -n, -max(alpha,m), depth+1);
10                IF t > m THEN
11                   IF (n = beta) OR (depth >= d-2)
12                   THEN m := t
13                   ELSE m := -negascout (p.i, -beta, -t, depth+1);
14                IF m >= beta THEN RETURN (m);
15                n := max (alpha,m) +1;
16          END;
17          RETURN (m);
18    END;
19 END;
```

Gambar 1. Pseudocode NegaScout

Jika terjadi *null window search fails high* (“t > m” di baris 10), *Negascout* harus mengunjungi kembali *subtree* yang sama dengan *wider window* untuk menunjukkan nilai *back-up* yang lebih baik. Pencarian dihentikan jika nilai *back-up* memiliki nilai sama dengan beta (“n = beta” di baris 11) atau saat *NegaScout* selalu mendapatkan nilai *minimax*

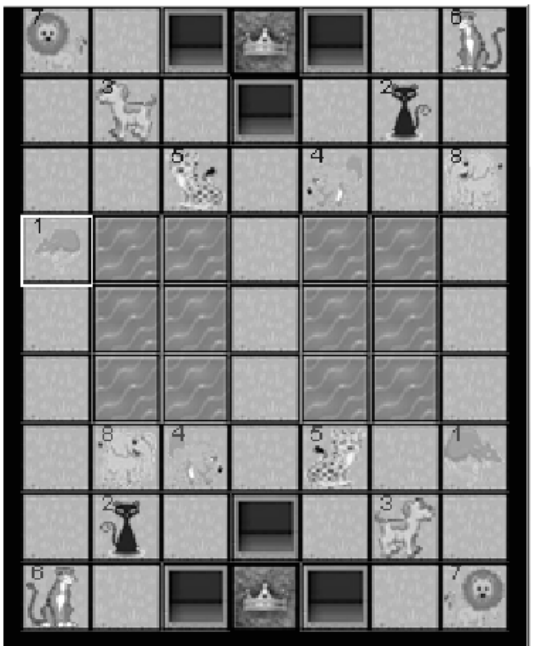
sampai dengan kedalaman minus dua ("depth >= d-2 "). Selain itu, pencarian harus diulang dengan *new window* (-beta, -t) (baris 13).

2.3. Animal Chess

Permainan *Animal Chess* adalah permainan tradisional dari China yang menggunakan papan permainan sebesar 7 x 9 kotak. Permainan ini dilakukan oleh dua orang. Masing–masing memiliki delapan buah bidak yang diwakili oleh binatang. Masing–masing binatang memiliki kelebihan dan kelemahannya tersendiri. Bidak–bidak tersebut antara lain adalah gajah, singa, harimau, macan tutul, serigala, anjing, kucing, dan tikus. Beberapa bidak memiliki kemampuan spesial, antara lain singa dan harimau dapat melompati sungai dengan syarat tidak ada hewan lain yang berada di dalam sungai. Tikus dapat berenang di air, dan tikus dapat menangkap gajah. (Burnett, 2010)

Aturan permainan dari *Animal Chess* antara lain:

a. Tujuan permainan adalah untuk menguasai sarang hewan milik lawannya dengan cara menempatkan bidak hewan ke dalam sarang tersebut.
b. Bidak hewan tidak dapat melangkah ke dalam sarangnya sendiri.
c. Masing- masing bidak bisa melangkah satu langkah secara vertikal maupun horisontal pada tiap giliran.
d. Masing–masing bidak dapat menangkap bidak yang memiliki nilai kurang dari atau sama dengan nilai yang dimilikinya, kecuali tikus yang bisa menangkap gajah dan gajah tidak dapat menangkap tikus.
e. Tikus dapat berenang di air dan melangkah satu langkah pada tiap gilirannya. Namun tikus tidak dapat menangkap gajah secara langsung dari air ke darat.
f. Tikus dapat menyerang tikus lawan dari dalam air ketika tikus lawan juga di dalam air.
g. Singa dan harimau dapat melompati sungai secara vertikal dan horisontal. Namun jika ada tikus di sungai yang menghalangi langkah singa dan harimau, singa dan harimau tidak dapat melompatinya.
h. Di sekitar sarang pemain terdapat jebakan dan bila hewan lawan menempati jebakan, hewan tersebut dapat ditangkap oleh semua hewan milik pemain tanpa memperdulikan nilainya.(Burnett, 2010)



Gambar 2. Papan permainan Animal Chess

2.3.1. Nilai Heuristik Animal Chess

Dalam penelitian ini, penulis menggunakan nilai heuristik yang akan digunakan oleh komputer cerdas dalam melakukan pencarian langkah. Nilai heuristiknya terdiri dari nilai langkah dan nilai posisi masing-masing bidak. Nilai heuristik tersebut dapat dilihat pada Gambar 2.5 sampai dengan Gambar 2.12 sebagai acuan untuk pemberian nilai langkah masing – masing bidak. (Burnett, 2010)

11	13	50	∞	50	13	13
11	12	13	50	13	13	13
10	11	11	13	13	13	13
8	9	9	11	12	12	13
8	9	9	11	12	12	12
8	9	9	10	12	12	11
8	8	8	9	10	10	10
8	8	8	9	9	9	9
8	8	8	0	8	8	8

*Gambar*3.Nilai langkah tikus.

11	15	50	∞	50	15	11
11	11	15	50	15	11	11
10	11	11	15	11	11	10
10	0	0	10	0	0	8
10	0	0	8	0	0	8
10	0	0	8	0	0	8
10	10	10	8	8	8	8
13	10	8	8	8	8	8
8	8	8	0	8	8	8

*Gambar*4. Nilai langkah kucing.

11	15	50	∞	50	15	11
10	11	15	50	15	11	10
9	10	11	15	11	10	9
9	0	0	10	0	0	9
8	0	0	8	0	0	8
8	0	0	8	0	0	8
8	8	10	8	8	8	8
8	12	13	8	8	8	8
8	12	12	0	8	8	8

*Gambar*5. Nilai langkah anjing.

11	15	50	∞	50	15	11
10	11	15	50	15	11	10
9	10	11	15	11	10	9
9	0	0	10	0	0	9
8	0	0	8	0	0	8
8	0	0	8	0	0	8
8	8	8	8	8	8	8
8	8	8	8	13	10	8
8	8	8	0	12	12	8

*Gambar*6. Nilai langkah serigala.

14	15	50	∞	50	15	14
13	14	15	50	15	14	13
13	13	14	15	14	13	13
12	0	0	15	0	0	12
11	0	0	14	0	0	11
10	0	0	13	0	0	10
9	9	9	10	10	9	9
9	9	9	9	9	9	9
9	9	9	0	9	9	9

*Gambar*7. Nilai langkah macan tutul.

25	30	50	∞	50	30	25
25	25	30	50	30	25	25
18	20	20	30	20	20	18
15	0	0	15	0	0	15
15	0	0	15	0	0	15
15	0	0	15	0	0	15
14	16	16	14	16	16	14
12	14	12	12	12	12	12
10	12	12	0	12	12	10

*Gambar*8.Nilai langkah harimau.

25	30	50	∞	50	30	25
25	25	30	50	30	25	25
18	20	20	30	20	20	18
15	0	0	15	0	0	15
15	0	0	15	0	0	15
15	0	0	15	0	0	15
14	16	16	14	16	16	14
12	12	12	12	12	14	12
10	12	12	0	12	12	10

*Gambar*9.Nilai langkah singa.

25	30	50	∞	50	30	25
25	25	30	50	30	25	25
18	20	20	30	20	20	18
16	0	0	16	0	0	16
14	0	0	14	0	0	14
12	0	0	12	0	0	12
10	15	14	14	14	14	12
11	11	11	11	11	11	11
11	11	11	0	11	11	11

*Gambar*10.Nilai langkah gajah.

Gambar 3 sampai Gambar 10 menunjukkan nilai dari masing-masing posisi yang akan didapatkan oleh masing-masing bidak ketika melangkah ke posisi yang sesuai dengan koordinatnya. Gambar tersebut mewakili nilai yang akan didapatkan oleh hewan *computer*. Dalam algoritma *negascout*, fungsi evaluasi tidak hanya digunakan untuk mencari nilai dari bidak *computer* (MAX), namun juga digunakan untuk mencari nilai terbaik dari langkah yang kemungkinan akan diambil lawan (MIN), dalam hal ini pemain, sehingga gambar tersebut juga digunakan untuk mencari nilai dari langkah bidak pemain, namun gambar yang digunakan menggunakan koordinat yang berlawanan dengan koordinat dari Gambar 3 sampai dengan Gambar 10. Nilai dari posisi awal bidak diberi nilai 10 dan nilainya akan terus bertambah saat mendekati sarang lawan karena hal ini adalah tujuan dari permainan, yaitu mendekati sarang lawan dan mengambil alih sarang mereka. Semua hewan kecuali tikus memiliki nilai posisi 0 di posisi air, karena hanya tikus yang dapat berenang di air. Kemudian, nilai 0 juga diberikan di posisi sarang MAX sehingga pemain MAX tidak akan melangkah ke dalam sarangnya sendiri. Selain itu, nilai *infinity* (∞) diberikan kepada MAX ketika MAX mencapai sarang MIN yang menyatakan bahwa sarang MIN merupakan *goal* dari permainan dan memiliki nilai paling besar. Tabel 1 menunjukkan nilai dari bidak.

Tabel 1.
Tabel nilai bidak.

Bidak	**Nilai**
Tikus	500
Kucing	200
Anjing	300
Serigala	400
Macan tutul	500
Harimau	800
Singa	900
Gajah	1000

Tabel 1 merupakan tabel dari nilai bidak MAX, nilai bidak dari MIN adalah nilai negatif dari nilai bidak MAX. Pemberian nilai terhadap nilai bidak lebih besar daripada nilai posisi karena dalam permainan ini, lebih bagus memakan bidak lawan daripada menempati sebuah posisi selain posisi sarang lawan. Tikus diberi nilai bidak 500 karena tikus memiliki kemampuan khusus yaitu dapat berenang di air. Sedangkan bidak lain diberi nilai yang sesuai dengan kemampuannya masing–masing.

3. Implementasi Sistem

Penulis menggunakan bahasa J2ME, karena J2ME sangat mendukung pembuatan aplikasi *mobile* yang digunakan dalam penelitian ini dalam bentuk aplikasi dua dimensi. Selain itu, penulis juga sudah terbiasa menggunakan Java. Sedangkan untuk pengkompilan *source code,* penulis menggunakan *batch script* dari Windows 7.

3.1. Proses Evaluasi

Dalam penerapannya, ketika *Computer* melakukan searching langkah menggunakan metode Negascout, *Computer* akan mencari nilai terbaik dengan mengasumsikan bahwa lawan juga akan selalu mencari posisi terbaik. Posisi terbaik tersebut dalam algoritma *Computer* disimbolkan dengan besar kecilnya nilai yang diperoleh seandainya pemain melangkah ke suatu posisi.

Dalam penelitian ini, peneliti mencari dari berbagai sumber dan meneliti secara langsung mengenai pemberian nilai langkah saat proses evaluasi dilakukan. Kemudian peneliti menggunakan nilai pada Gambar 3 sampai dengan Gambar 10 sebagai acuan untuk pemberian nilai terhadap langkah masing – masing bidak. Selain mencari nilai dari posisi, fungsi evaluasi juga menentukan nilai dari sebuah bidak yang dapat dimakan, baik oleh MAX maupun MIN. Tabel 1 menunjukkan nilai dari bidak.

Pemberian nilai terhadap nilai bidak lebih besar daripada nilai posisi karena dalam permainan ini, lebih bagus memakan bidak lawan daripada menempati sebuah posisi selain posisi sarang lawan. Tikus diberi nilai bidak 500 karena tikus memiliki kemampuan khusus yaitu dapat berenang di air. Sedangkan bidak lain diberi nilai yang sesuai dengan kemampuannya masing–masing.

3.2. Implementasi Algoritma NegaScout

Algoritma *Negascout* menggunakan *parameter* giliran, *alpha*, *beta* dan kedalaman. Sedangkan dalam algoritma *negascout* itu juga terdapat beberapa kondisi yang terkait dengan algortima *negascout*. Kondisi–kondisi tersebut antara lain:

a. Jika kedalaman sudah sama dengan 0, maka pencarian dihentikan dan fungsi evaluasi dipanggil.
b. Nilai terbaik sementara diinisialisasikan dengan nilai negatif maksimum.
c. Vector dari *successor* merupakan node–node langkah yang dapat diambil oleh giliran pemain yang melangkah.
d. Diasumsikan bahwa nilai terbaik adalah nilai dari node pertama, sehingga langkah awal yang diambil adalah langkah dari node pertama hingga ditemukan ada nilai yang lebih baik daripada nilai dari node pertama.
e. Pencarian terhadap suatu cabang *tree* akan dihentikan jika tidak ditemukan nilai yang lebih baik dari node pada cabang lain yang sudah dicari sebelumnya.
f. Jika ditemukan nilai yang lebih baik, maka dilakukan pencarian ulang pada cabang node tersebut.
g. Penegasian dari nilai fungsi negascout menandakan adanya minimalisasi langkah dari MIN terhadap nilai langkah MAX.
h. Algoritma negascout mengembalikan nilai terbaik yang didapatkan pada waktu pencarian dalam sebuah node.

Kedalaman papan yang digunakan adalah kelipatan dari 2, karena fungsi negascout menghitung nilai MAX setelah MIN melangkah. Sedangkan langkah dimulai dari MAX pada kedalaman 0.

Gambar 11 merupakan gambar potongan code dari *method* negaScout yang telah diimplementasi oleh penulis.

```
14      public int negaScout(boolean isAI, int alpha, int beta, int depth)
15      {
16          if(depth==0)
17              return estimate();
18          int best = -Constants.MAX_VALUE;
19          int n = beta;
20          Vector v = successors(isAI);
21          if(v!=null)
22          {
23              int siz = v.size();
24              v = randomize(board.a, v);
25              if(v.size()>0 && depth==dd)
26                  mm = (Move)v.elementAt(0);
27              while(v.size()>0 && best<beta)
28              {
29                  Move m = (Move)v.elementAt(0);
30                  v.removeElementAt(0);
31                  m.perform(board);
32                  int est = -negaScout(!isAI, -n, -Math.max(alpha, best), depth-1);
33                  if(est>best)
34                  {
35                      if(n==beta || depth<=2)
36                      {
37                          best = est;
38                          if(depth==dd)
39                              mm = m;
40                      }
41                      else
42                      {
43                          best = -negaScout(!isAI, -beta, -est, depth-1);
44                          if(depth==dd)
45                              mm = m;
46                      }
47                  }
48                  m.undo(board);
49                  n = Math.max(alpha, best)+1;
50              }
51          }
52          return best;
53      }
54  }
```

Gambar 11. Method NegaScout.

4. Pengujian Sistem

Dalam penelitian ini, penulis akan melakukan penelitian dalam berbagai aspek mengenai efektivitas dan efisiensi algoritma *negascout* dibandingkan dengan algoritma *alpha-beta* dan algoritma *Minimax* pada permainan *AnimalChess*.

Penelitian ini dibagi dalam 3 aspek, yaitu:

a. Aspek waktu.
   Merupakan lama waktu yang dibutuhkan oleh masing-masing algoritma dalam menemukan langkah terbaik untuk diambil. Lama waktu dihitung dengan satuan mili detik. Dengan cara menghitung selisih waktu mulai dengan waktu berhentinya pencarian.
b. Aspek jumlah node.
   Merupakan jumlah node yang dihasilkan oleh masing-masing algoritma hingga menemukan langkah terbaik dan pencarian dihentikan. Satuan yang digunakan adalah jumlah node dalam *integer*.
c. Aspek presentase kemenangan.
   Merupakan presentase kemenangan komputer melawan manusia menggunakan masing-masing algoritma. Satuan yang digunakan berupa persen.

Penelitian dilakukan dengan *device* yang sama, yaitu Sony-ericsson k800i. Output ditulis dengan method drawString langsung ke dalam layar device.

4.1.Hasil Perbandingan Negascout dan Alpha-Beta

Dalam evaluasi, penulis melakukan pengujian sistem dengan kedalaman dua dan kedalaman empat. Hal ini dikarenakan kemampuan perangkat telepon seluler yang terbatas untuk pengujian. Pengujian dilakukan dengan kondisi langkah yang sama untuk masing-masing algoritma.

Melalui Tabel 2, penulis menyimpulkan bahwa *Negascout* memerlukan waktu lebih singkat dalam pencarian node karena jumlah node yang ditelusuri oleh *Negascout* lebih sedikit daripada *Alpha-Beta*pada kedalaman empat. Selain itu, penulis juga menyimpulkan

bahwa pada kedalaman dua Negascout dan Alpha-Beta dapat menemukan pencarian yang menghasilkan jumlah node yang sama.

Dengan demikian, penulis menyimpulkan bahwa algoritma Negascout dapat mencari dan menemukan node terbaik untuk melangkah lebih efektif dan lebih efisien jika diterapkan dengan kedalaman lebih dari sama dengan empat. Sedangkan untuk kedalaman kurang dari sama dengan dua, penulis menyimpulkan bahwa *Negascout* dan *Alpha-Beta* memiliki keefektifan yang sama dalam hal penelusuran jumlah node.

Penulis melakukan penelitian mengenai presentase kemenangan manusia melawan komputer cerdas dalam permainan *AnimalChess* dengan membandingkan algoritma Negascout dengan algoritma Alpha-Beta pada kedalaman empat yang dimainkan menggunakan telepon selular merk Sony-ericsson k800i.

Tabel 2.
Tabel hasil pengujian jumlah node dan waktu menggunakan telepon selular.

Kedalaman Pencarian	Langkah Bidak ke -	Negascout		Alpha-Beta	
		Jumlah Node	Waktu Pencarian	Jumlah Node	Waktu Pencarian
2	1	71	1066	71	1066
	2	237	1548	237	1557
	3	117	1174	117	1185
	4	254	1681	254	1692
	5	162	1330	162	1317
4	1	2837	18873	2905	19912
	2	5408	29470	6113	34116
	3	3668	22007	4037	25818
	4	5266	30582	5386	33684
	5	3724	24032	3887	24781

Melalui Tabel 3, penulis menyimpulkan bahwa presentase kemenangan manusia melawan komputer dalam permainan *AnimalChess* menggunakan algoritma *Negascout* lebih besar daripada menggunakan algoritma *Alpha-Beta*.

Presentase kemenangan tersebut dihasilkan dari percobaan sepuluh orang teman penulis termasuk penulis untuk melawan komputer menggunakan masing-masing algoritma. Namun, faktor pengetahuan pemain tentang permainan *AnimalChess* juga berpengaruh terhadap hasil presentase kemenangan tersebut karena kebanyakan sampel pemain belum mengetahui aturan dan cara bermain permainan *AnimalChess* menyebabkan beberapa pemain kalah melawan komputer.

Tabel 3.
Tabel hasil pengujian kemenangan menggunakan telepon dengan kedalaman empat.

No	Negascout	Alpha-Beta
1	Pemain menang	Pemain menang
2	Pemain menang	Komputer menang
3	Komputer menang	Pemain menang
4	Pemain menang	Komputer menang
5	Komputer menang	Komputer menang
6	Pemain menang	Pemain menang
7	Komputer menang	Komputer menang
8	Komputer menang	Pemain menang
9	Komputer menang	Komputer menang
10	Komputer menang	Pemain menang
Presentase Kemenangan Komputer	60%	50%

4.2. Hasil Perbandingan Negascout dan Minimax

Penulis melakukan pengujian sistem membandingkan algoritma *Negascout* dan *Minimax* dengan kedalaman dua. Hal ini dikarenakan kemampuan perangkat telepon yang terbatas untuk pengujian karena algoritma *Minimax* mencari semua node yang dapat dihasilkan. Pengujian dilakukan dengan kondisi langkah yang sama untuk masing-masing algoritma.

Melalui tabel 4 penulis menyimpulkan bahwa algoritma *Minimax* memerlukan waktu yang lebih lama dalam pencarian sebuah node terbaik dan menghasilkan jumlah node yang lebih banyak daripada algoritma *Negascout*. Hal ini dikarenakan algoritma *Minimax* mencari semua kemungkinan langkah yang dapat dihasilkan oleh masing-masing pemain dengan kedalaman tertentu yang diasumsikan bahwa masing-masing pemain akan mengambil langkah yang terbaik.

Tabel 4.
Tabel hasil pengujian jumlah node dan waktu menggunakan telepon selular.

No	Negascout		Minimax	
	Node	Time (ms)	Node	Time (ms)
1	71	1066	576	1365
2	237	1548	480	2238
3	117	1174	480	2347
4	254	1681	457	2349
5	162	1330	436	2281

4.3. Kendala Penelitian

Dalam penelitian ini, penulis mengalami dua kendala dalam melakukan penelitian. Diantaranya adalah:

a. Ketika penulis akan meneliti presentase kemenangan antara manusia melawan komputer yang menggunakan algoritma *Minimax*, penulis mencoba untuk memberikan kedalaman lebih dari dua kepada komputer untuk melakukan pencarian. Hal ini mengakibatkan komputer berpikir terlalu lama dan membuat permainan menjadi terhenti saat komputer mencari langkah. Hal ini membuat peneliti tidak dapat meneliti jumlah node dan waktu yang diperlukan untuk komputer dapat mengambil langkah pada algoritma *minimax* dengan kedalaman lebih dari dua.
b. Dengan maksimum kedalaman dua, komputer yang menggunakan algoritma *minimax* menjadi sangat mudah untuk dikalahkan oleh manusia sehingga peneliti memutuskan untuk tidak melakukan penelitian mengenai presentase kemenangan manusia melawan komputer menggunakan algoritma *minimax* dengan kedalaman dua.

5. Kesimpulan

Dalam permainan *AnimalChess*, algoritma *Negascout* memerlukan waktu yang lebih sedikit dan memotong lebih banyak node dibandingkan algoritma *Alpha-Beta* pada kedalaman 4. Selain itu, presentase kemenangan komputer yang menggunakan algoritma *Negascout* dalam permainan *AnimalChess* sedikit lebih besar daripada presentase kemenangan komputer yang menggunakan algoritma *Alpha-Beta*. Pada akhirnya dari penelitian ini menunjukkan bahwa algoritma *Negascout* jauh lebih cepat menemukan node langkah terbaik dibandingkan dengan algoritma *Minimax*.

Daftar Pustaka

Adams, E., & Rollings, A. (2007). *Fundamentals of Game Design.* New Jersey: Prentice Hall.

Burnett, J. (2010). *Discovering and Searching Loosely Coup.* Medford: Tufts University.

Chang, H.-J., Tsai, M.-T., & Hsu, T.-s. (2003). *Game Tree Search with Adaptive Resolution.* Taipei: Academia Sinica.

Luger. (2011, 9 7). *CPSC 352 -- Artificial Intelligence -- Class Notes 2011*. Retrieved 8 26, 2012, from Trinity College: http://www.cs.trincoll.edu/~ram/cpsc352/notes/

Reinefeld, A. (1983). An Improvement to The Scout Tree Search Algorithm. *ICCA Journal Volume 6 No 4* , 4-14.

Russell, S. J., & Norvig, P. (1995). *Artificial Intelligence A Modern Approach.* New Jersey: Prentice Hall.

Schaeffer, J., & Herik, J. V. (2002). *Games, Computers, and Articial Intelligence, Articial Intelligence.* Amsterdam: Elsevier.